**PANOSETI Quabo Firmware Release #7**

RR August 12, 2019

This release is for the “stand-alone” (no Mobo) Quabo. It implements the White Rabbit time distribution system. The FPGA configuration file is quabo\_v007.mcs.

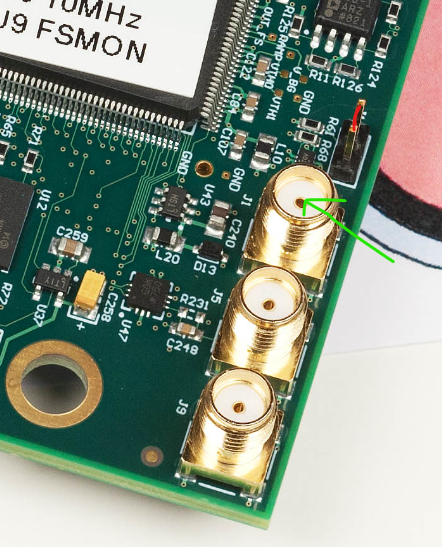
The system will work without connection to a WR system but the elapsed time will not be synchronized to any other board.

To use the WR system, you need to use the correct fiber transceiver, and connect to an upstream WR node set to “Master” or “Grandmaster” (the latter being the node which accepts 10MHz and 1PPS timing signals from the primary GPS-trained source). The transceivers that I’ve used are as follows:

In Quabo AXGE-1254-0531

In WR-LEN AXGE-3454-0531

The 1PPS output is available on the SMA J1 (green arrow); the adjacent jumper J11 (red line) needs to be **installed** to make this an output (we had previously used it as the 1PPS input):

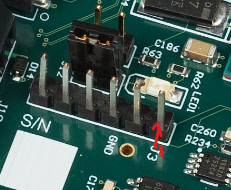


This SMA is terminated on the PCB; terminate it in the scope also for a clean 1.5v pulse.



Magenta is the quabo 1PPS output, green the WR\_LEN.

The J3 testpoint block now has the following functions:



Pins 1 & 2: install jumper for IP address 192.168.1.10, remove for 192.168.1.11 (as before)

Pin 3: WR UART Tx (output from board)- see below

Pin 4: WR UART Rx

Pin 5: Stim\_trigger (signal synchronous with stim generator trigger)

Pin 6: Any\_trigger (the OR of all 256 MAROC trigger outputs)

In making these changes to two boards I found that the U9 buffer on Quabo SN04 was installed backwards- recall that we also had that problem on a board that Jerome has. And I can see that the part is backwards on SN03, which Paul photographed, too. Here’s how it shouldn’t be:



That little part behind the SMA with the “TQV” is wrong-way around.

**The WR Shell**

WR has a shell command/status interface. This is accessible either through a UART connection to J3 or through the Ethernet connection (I’ve included both in case the Ethernet access has problems; also the WR system is active before the MicroBlaze, so the WR boot activity is visible on the UART output before the MB wakes up). You can select which connection is used with a new parameter in the configuration file:



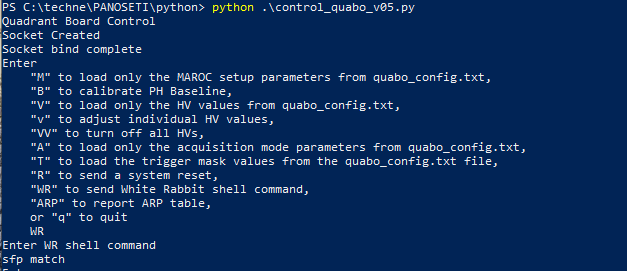
The default value of EN\_WR\_UART is 0, so that the system starts up with the Ethernet connection active. To use the UART connection instead, see below.

The shell commands are described in the White Rabbit PTP Core User’s Manual:

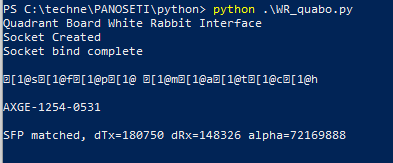
https://www.ohwr.org/attachments/5299/wrpc-user-manual-v4.1.pdf

**User Interface**

I’ve modified the quabo\_control program to allow writing WR commands; new version is control\_quabo\_v05.py (Jerome can incorporate these changes to his setup code easily). I’ve also written a separate program which receives the WR status output from the shell; I’ve called this WR\_quabo\_v01.py (the output is sent to a separate UDP port so a separate process can display it). Here’s an example of a WR command “sfp match” sent from the control program, and the resultant display in WR\_quabo\_v01.py:



*Enter ‘WR’ then “sfp match”*



*Response*

Note that the first line of the response has a bunch of rubbish characters- in fact this is just the echo of the “sfp match” but with a bunch of interleaved non-printing characters, which perhaps would give the text color or italics or something with the proper terminal emulator. I tried to strip these out in Python but couldn’t find the right formula.

The second and third lines are the response, showing the transceiver name and parameters.

The shell command “gui” causes the system to spit out a status packet every second; to stop this one needs to send an “escape” character. You can do this by entering a carriage return after the “Enter WR shell command” prompt.

**Setting up the WR system**

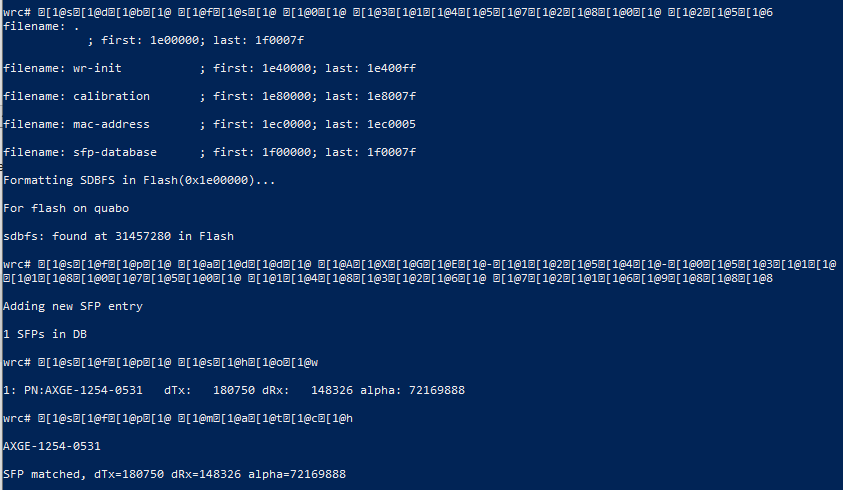
After programming the config file and booting the system, the WR system will be running. But the filesystem (stored in the flash RAM) is empty. This needs to be set up by entering some commands into the WR shell:

sdb fs 0 31457280 256

{that number is 0x1e00000, but it doesn’t seem you can enter it in hex}

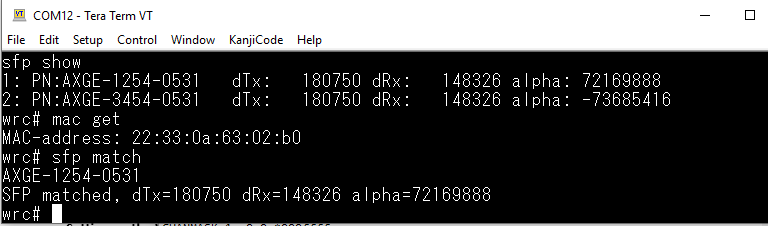
sfp add AXGE-1254-0531 180750 148326 72169888

Then you can do “sfp show” and “sfp match” to show the contents of the sfp transceiver database and to see that the correct transceiver is installed. The output in response to all of this is shown here:



**Using the UART interface to the WR shell**

You can also send and receive WR shell data using a UART connected to the two pins of J3. Use a terminal program such as Tera Term. Set the UART to 115,200kb, 1 stop. Change the quabo\_config.txt parameter EN\_WR\_UART to 1, and send the ‘A’ (send acquisition parameters) command in control\_quabo\_v05.py. Then you can type commands and see the response in the terminal window:



This can also be useful to see the WR activity during boot-up, before the MicroBlaze system is active. Also, the Ethernet response requires the MicroBlaze processor to be interrupted for every character sent by the WR shell, which busies up the processor somewhat. Note that the WR output can be seen at the UART even when using the Ethernet WR interface, since there is no reason to turn this off.

**Other changes**

The elapsed time clock tick is now 3ns, rather than 3.125 (333.333 MHz is easily made from the 62.5MHz WR clock)

The MicroBlaze clock was slowed from 100MHz to the 62.5MHz WR clock to keep things synchronous, slowing the software speed accordingly (so the max frame rate in PH mode is likely less than before). We can maybe double this, but I wanted to keep it the same for now.

The stim rate and frame rate for image mode have changed accordingly (reduced by a factor of 10ns/16ns = 5/8)

I fixed a bug wherein most of the housekeeping data was really only updated every 8 seconds- the data were repeated in pairs. Now the housekeeping produces new data every 2.7s (the 16-channel delta-sigma HK ADC has a max 150ms conversion time; .15s \* 16 = 2.4s)